
Introduction aux ontologies du Web Sémantique : technologies et outils

Séminaire équipe linguistique, LabInfo IGM

27 avril 2009

Organisation

1. Web sémantique
2. Les technologies: Logiques de descriptions, RDF, OWL
3. Les outils
 - Éditeur : Protégé
 - Framework Java : Jena
 - Raisonneur : Pellet
 - Language de requête : SPARQL
 - "Triple Store" : Sesame

[1]

Le web sémantique

Web syntaxique

- Le web actuel est syntaxique.
 - On va écrire en utilisant du HTML (voire XHTML) des pages qui gèrent la présentation du document.
 - Mais rien sur la sémantique du contenu.
- Conséquence : le système informatique prend en charge la présentation (facile) et l'humain interprète les informations (difficile).

Exemple

- Le balisage permet :
 - La présentation des informations (polices, couleurs, etc.).
 - Établir des liens hypertextuels
- La sémantique du contenu est accessible aux humains mais c'est une tâche difficile pour les ordinateurs.



Nous pouvons voir ...

WWW2002

The eleventh international world wide web conference

Sheraton waikiki hotel

Honolulu, hawaii, USA

7-11 may 2002

1 location 5 days learn interact

Registered participants coming from

australia, canada, chile denmark, france, germany, ghana, hong kong, india, ireland, italy, japan, malta, new zealand, the netherlands, norway, singapore, switzerland, the united kingdom, the united states, vietnam, zaire

Register now ...

La machine va voir

[illegible]

Solution : des balises XML

<name>






<location>•⋈ℓ□◻◆□■ •◻)(&)(&)(⋈□◆ℓ●



<date>   </date>

<slogan> **</slogan>**

<participants> 

[illegible]

<introduction> ☉ ㊦ ㊧ ㊨ ㊩ ㊪ ㊫ ㊬ ㊭ ㊮ ㊯ ㊰ ㊱ ㊲ ㊳ ㊴ ㊵ ㊶ ㊷ ㊸ ㊹ ㊺ ㊻ ㊼ ㊽ ㊾ ㊿

၂။

●□■&■□• ▯▮×)(□○▮ ≡

<speaker>⊗⊗○ ⊗⊗□■⊗□•⊗●⊗⊗</speaker>

La machine va maintenant voir

<■□○■>◆◆◆■□□■
■■■■ ■■●■◆■ ■■◆■□■□■◆■□■□■ ◆■□■◆ ◆■□■◆ ■■●■◆ ■■●■◆</■□○■>
<●□■□◆◆■□■>◆■■■□■□■ ◆■(■)(■)(■) ■■◆■●
■□■□◆◆◆◆ ■■◆■(■)(■) ◆◆◆</●□■□◆◆■□■>
<◆□■◆>■◆◆◆ ○◆■ ■□□■</◆□■◆>
<◆●□■□■>■ ●□■□◆■□■ ■ ◆□■◆ ◆■□■ ◆■◆■□■□■◆</◆●□■□■>
<□□□◆■□■□■◆>■□■□■◆■□■□■ □□□◆■□■□■◆ ■□○■□■ □□□
◆◆◆□■□■□■ □■□■□■ □■□■□■ □■□○□■◆ □■□■□■
■□□○□■□■ □■□■□■ ■□■□■ ◆■□■□■ ◆■□■□■ □■□■□■
◆◆◆□■□■ □■□■□■ □■□■□■ ■■◆ ■■□■□■□■ ◆■■■
■■◆■■■□■□■□■ □■□■□■ ◆■□■□■□■ □■□■□■ □■□■□■
◆■■■ ◆■□■□■ ◆■□■□■ □■□■□■ ◆■■■ ◆■□■□■ ◆■□■□■
◆■□■□■ □■□■□■ ◆■□■□■</□□□◆■□■□■◆>
<■◆■□□■◆■□■>■□■□■◆■□■ ■□■
■ ◆■■■ ■◆◆ ◆□■ ■□□□◆◆ ◆■□■ □□□◆■□■ ◆■■■
■□■□■□■ □■ ◆■■■ ■■□■◆■ □■□■◆■ □■□■◆ ■□■◆
◆■□■ □■□■□■□■□■</■◆■□□■◆■□■>
<◆□■□■□■>■□○ ■□■□■□■◆■□■</◆□■□■□■>
<■□■>■□○ ◆■ □■□■ ◆■□■ ◆■□■□■ □■ ◆■■■ ◆</■□■>
<◆□■□■□■>■□■ □■◆■□■</◆□■□■□■>

Ajouter de la sémantique

- Ajouter des annotations
- En exploitant des technologies du domaine de la représentation des connaissances pour spécifier le sens des annotations.
 - Utilisation d'une syntaxe compatible avec le Web
=> XML.
 - En proposant des méthodes pour l'inférence de nouvelles informations / connaissances.

Une extension du Web actuel

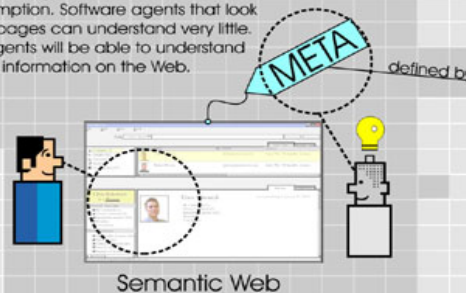
- "The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation." [1].
- Comment y parvenir ?
 - À l'aide de structures XML
 - En développant des ontologies.

[1] Tim Berners-Lee, James Hendler, Ora Lassila, The Semantic Web, Scientific American, May 2001



3 What was confusing, now makes perfect sense.

Currently, a Web page is developed mainly for human consumption. Software agents that look at the current pages can understand very little. In the future agents will be able to understand the majority of information on the Web.



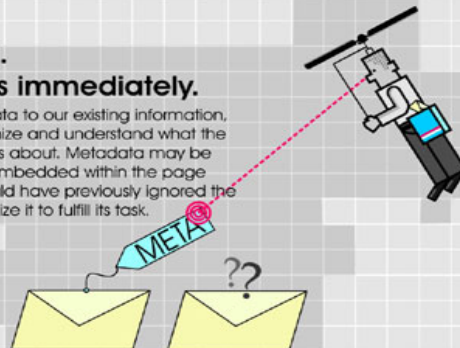
4 Ontologies give the metadata meaning.

A Semantic Web page consists of RDF and linkages to global ontologies. Agents can understand the RDF associated with a page once they have crawled the web of ontologies. It is this set of ontologies or vocabularies that act as a dictionary of sorts to the agents who are attempting to understand the terms used to describe the page. This web of ontologies exists in parallel with the Semantic Web.



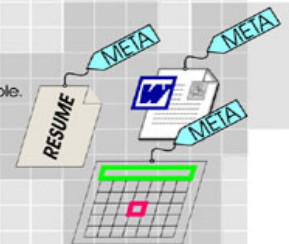
2 Add metadata. Serve to agents immediately.

By attaching additional data to our existing information, agents are able to recognize and understand what the information is and what it is about. Metadata may be linked to a web page or embedded within the page itself. Where an agent would have previously ignored the information, it can now utilize it to fulfill its task.



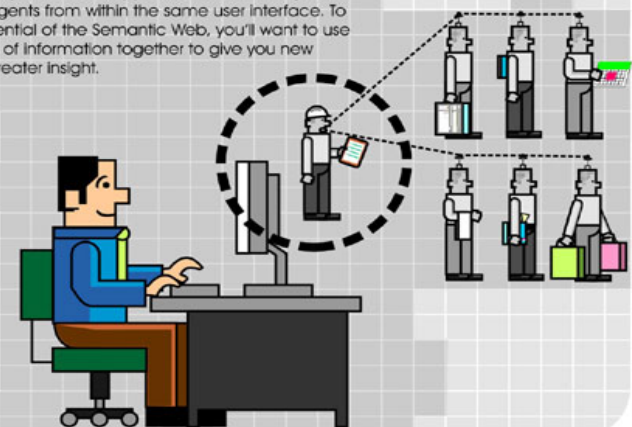
5 The Semantic Web isn't just about Web sites either!

The Semantic Web aims to make many different types of data usable. Emails, Web sites, calendars, resumes, office documents, and contacts are just some of the information resources that can be utilized by software agents if they are enhanced with metadata. Imagine what new applications are possible once agents can tell the difference between these items!



6 How will I use the Semantic Web?

In all likelihood, you'll use the Semantic Web using many different applications. You may have particular applications that use a specific type of information, like calendaring for example. Alternatively, you may choose an application that allows you to control a bevy of agents from within the same user interface. To harness the full potential of the Semantic Web, you'll want to use the different islands of information together to give you new applications and greater insight.



1 It's evolution not revolution.

The Semantic Web leverages the content and services that already exist in the World Wide Web. By extending the World Wide Web to be more usable to robots, agents or machines, new applications are now possible.

- ← The Semantic Web is a layer of machine understandable metadata on top of the World Wide Web
- ← The World Wide Web is the foundation for the emerging Semantic Web

XML n'est pas suffisant

- XML est comme HTML mais il permet de définir ses propres balises.
- Cependant, avec XML, on ne peut pas exprimer le sens des balises.
- C'est le rôle des ontologies qui :
 - expriment la sémantique dans un formalisme interprétable par les machines.
 - permettent à des programmes indépendants d'échanger des données / connaissances.

Ontologie

- Une branche de la philosophie s'occupant de la nature et de l'organisation de la réalité.
 - Etude de l'être en tant qu'étant.
- En science l'information (et IA) : définir les concepts et les relations entre concepts pour un domaine défini.

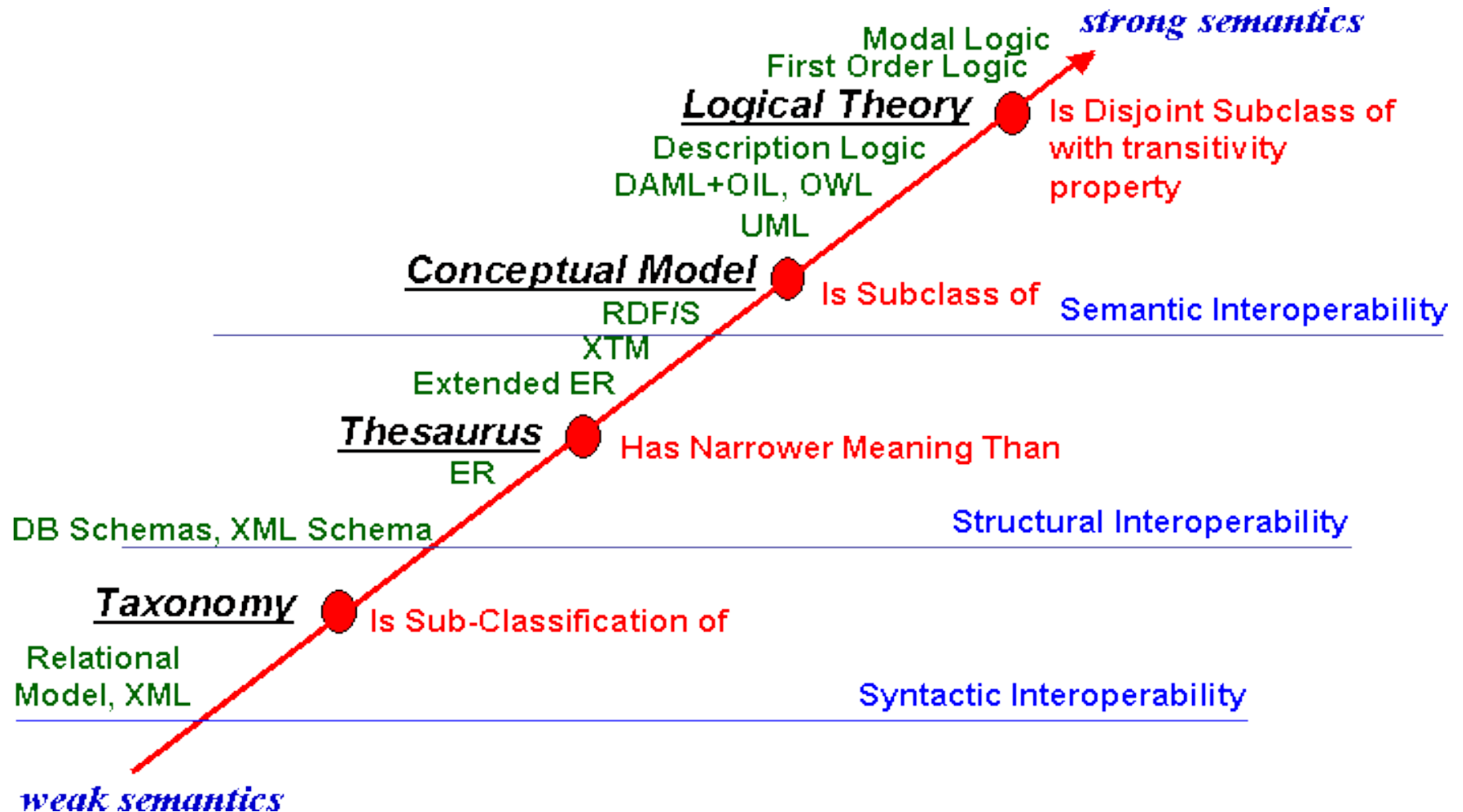
Définition de Gruber

- "An ontology is a **formal, explicit specification** of a **shared conceptualization**" [2].
- Explications
 - Formal : formaliser une ontologie de manière à la rendre "traitable" pour une machine.
 - Explicit specification : pour exprimer que les concepts et les contraintes sont explicitement définis avec une forme concrète.
 - Shared : une ontologie capture des connaissances consensuelles.
 - Conceptualization : se réfère à un modèle abstrait portant sur des phénomènes du monde. Ce modèle présente les concepts pertinents de ce phénomène.

Les types d'ontologies

- Pour Guarino, il existe 3 types :
 - **Top-level ontologies** : Elles décrivent des concepts généraux et indépendants d'un domaine d'application particulier. Exemples : WordNet et Cyc
 - **Domain and tasks ontologies** : Elles décrivent respectivement les concepts pour un domaine et une tâche générique.
 - **Application ontologies** : descriptions des concepts exploités dans une application, donc dépendance du domaine et de la tâche.
- Mais il existe d'autres classifications (Fensel, Studer, etc..)

Spectre des ontologies



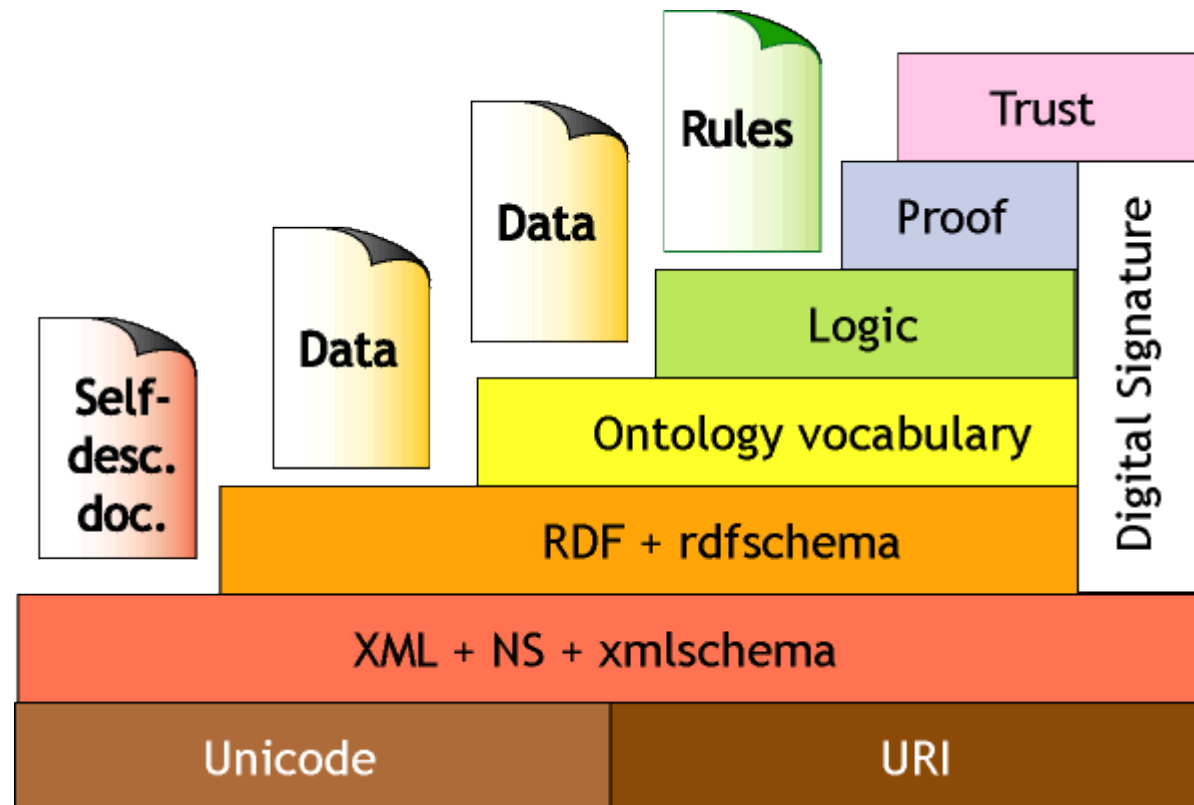
Exemples

- Wordnet
- Dublin Core
- Cyc
- BioInformatique: GO (Gene Ontology), OBO, Tambis, ..
- Médical: Galen, SNOMED, ATC, EphMRA, ..
- Géographique: Corine, ATKIS, Geoland,...

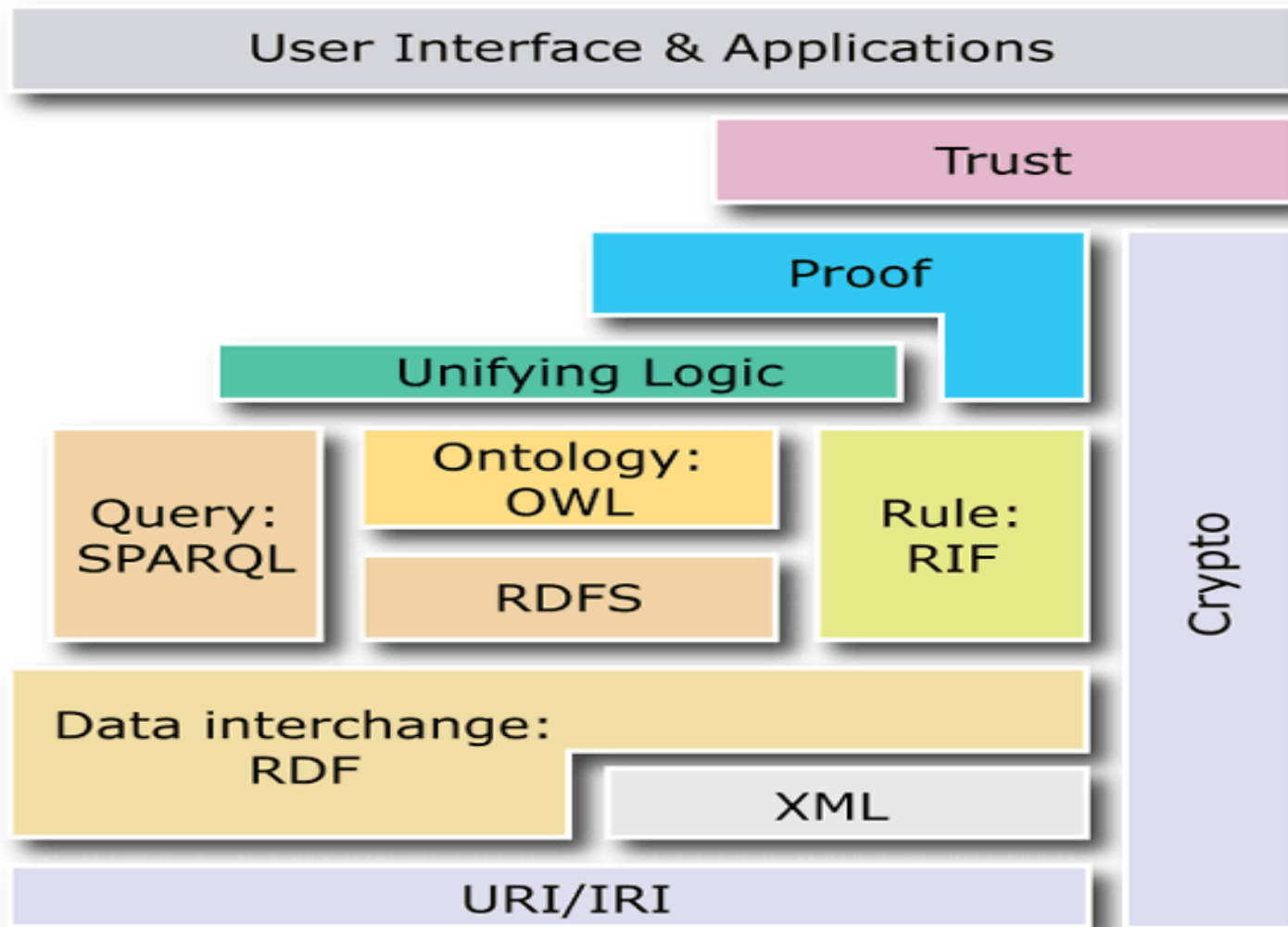
[2]

Les technologies

Une architecture du web sémantique



La plus récente



[2.1]

RDF et RDFS



Présentation de RDF

(Resource Description Framework)

- Un modèle de données pour décrire les ressources du Web
- Supporte une interopérabilité entre applications partageant des informations (interprétables par les machines) sur le Web.
- Possède une syntaxe XML (mais ce n'est pas l'unique syntaxe).

RDF, pour faire quoi ?

- Amélioration de la découverte de ressources sur le Web.
- Etablir des catalogues de ressources (décrire le contenu et les relations).
- Développement d'agents intelligents.
- Spécifier la sémantique des données d'un document XML.

Modèle de données RDF

- Equivalent aux réseaux sémantiques [Staab 2000] : graphe orienté avec labels.
- Les noeuds représentent des concepts, des instances et les valeurs des propriétés.
- Les arcs représentent des propriétés entre concepts.

RDF triplet

- Une déclaration RDF est un triplet constitué d'un sujet, d'une propriété et d'un objet.
- D'un sujet (resource)
 - Un objet du domaine
 - Pointé par un identifiant (URI)
- D'une propriété (predicat – relation)
 - Relation binaire sur le domaine entre un sujet et un objet.
 - Également une ressource (URI).
- D'un objet

Concepts clés de RDF

- Modèle les déclarations sous forme d'un graphe (orienté labellisé).
- Exploite les URI, espace de noms
- Possibilité de typer les données
- Proposer des littéraux
- Plusieurs présentations (XML, N3, N-triples, graphique).
- Exprime des faits simples

Sérialisation XML/RDF

- Code XML/RDF :

```
<?xml version="1.0"?>
```

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"

```

```
  xmlns:dc="http://purl.org/dc/elements/1.1/">
```

```
  <rdf:Description rdf:about="http://www.univ-mlv.fr/~ocure">
```

```
    <dc:title>Page de olivier Cure</dc:title>
```

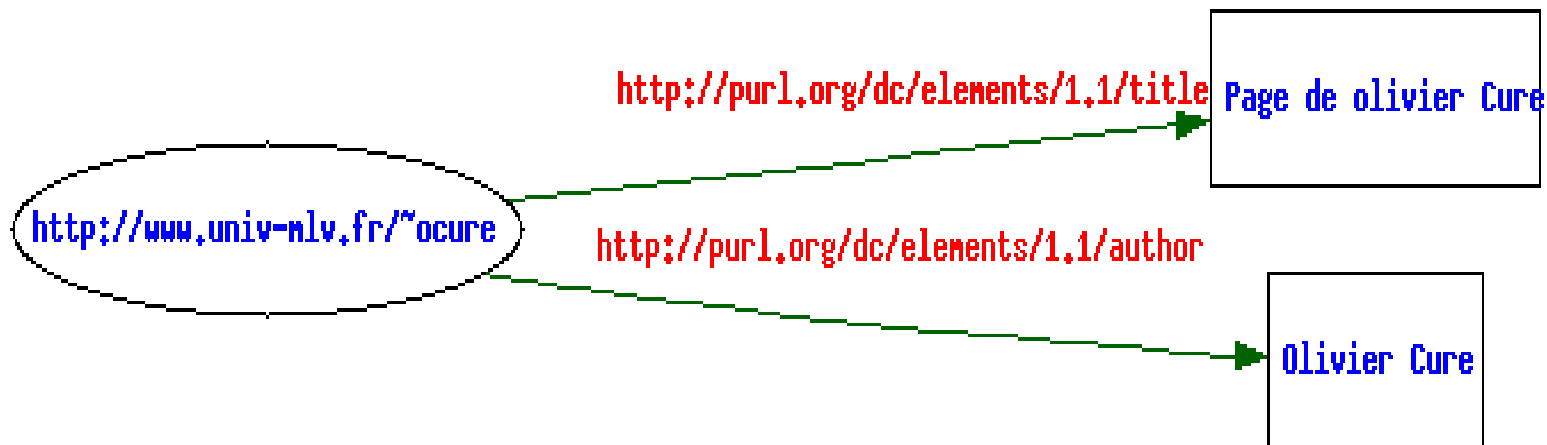
```
    <dc:author>Olivier Cure</dc:author>
```

```
  </rdf:Description>
```

```
</rdf:RDF>
```

Représentation graphique

Noeud : sujet et objet
arc : prédicat



Formats N-triples et N3

- Format N-triples

```
<http://www.univ-mlv.fr/~ocure>  
  <http://purl.org/dc/elements/1.1/title> "Page de  
  olivier Cure" .
```

```
<http://www.univ-mlv.fr/~ocure>  
  <http://purl.org/dc/elements/1.1/author> "Olivier  
  Cure" .
```

- Format N3

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .  
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-  
syntax-ns#> .
```

```
<http://www.univ-mlv.fr/~ocure>  
  dc:author "Olivier Cure";  
  dc:title "Page de olivier Cure" .
```

RDF Schema (RDFS)

- Un langage pour décrire des langages RDF.
- Permet de définir les types de ressources (personne, livre, etc.) ainsi que leurs propriétés (diplôme, titre, auteur, etc.).
- RDFS propose de l'information sur l'interprétation des déclarations RDF.

RDFS (2)

- Etendre RDF à la description d'ontologies.
 - Hiérarchies de classes et propriétés :
 - SubClassOf, subPropertyOf
 - range, domain sur les propriétés.
 - Annotations
 - seeAlso, isDefinedBy, label, range, domain, member
 - Préfixe : “rdfs” et URI :”http://www.w3.org/2000/01/rdf-schema#”

Les primitives de RDFS (propriétés)

Property name	comment	domain	range
rdf:type	The subject is an instance of a class.	rdfs:Resource	rdfs:Class
rdfs:subClassOf	The subject is a subclass of a class.	rdfs:Class	rdfs:Class
rdfs:subPropertyOf	The subject is a subproperty of a property.	rdf:Property	rdf:Property
rdfs:domain	A domain of the subject property.	rdf:Property	rdfs:Class
rdfs:range	A range of the subject property.	rdf:Property	rdfs:Class
rdfs:label	A human-readable name for the subject.	rdfs:Resource	rdfs:Literal
rdfs:comment	A description of the subject resource.	rdfs:Resource	rdfs:Literal
rdfs:member	A member of the subject resource.	rdfs:Resource	rdfs:Resource
rdf:first	The first item in the subject RDF list.	rdf:List	rdfs:Resource
rdf:rest	The rest of the subject RDF list after the first item.	rdf:List	rdf:List
rdfs:seeAlso	Further information about the subject resource.	rdfs:Resource	rdfs:Resource
rdfs:isDefinedBy	The definition of the subject resource.	rdfs:Resource	rdfs:Resource
rdf:value	Idiomatic property used for structured values (see the RDF Primer for an example of its usage).	rdfs:Resource	rdfs:Resource
rdf:subject	The subject of the subject RDF statement.	rdf:Statement	rdfs:Resource
rdf:predicate	The predicate of the subject RDF statement.	rdf:Statement	rdfs:Resource
rdf:object	The object of the subject RDF statement.	rdf:Statement	rdfs:Resource

Constats sur RDF(S)

- Puissance expressive insuffisante, il manque :
 - Cardinalités (min et max)
 - Décomposition (disjoint, exhaustivité).
 - Axiomes
 - Négation
- Ne permet pas de tester la consistance de la KB

RDF/RDFS problèmes

- Pas de distinctions entre classes et instances

`<Espece, type, Class>`

`<Lion, type, Espece>`

`<Simba, type, Lion>`

- Les propriétés peuvent avoir des propriétés
- Pas de distinctions entre constructeurs du langage et les termes de l'ontologie.

Solution 2 : RDF (1)

```
<?xml version="1.0"?>
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns="http://www.univ-mlv.fr/~ocure/etudSchema.rdf#">

    <Cours rdf:ID="Cours1">
      <possedeNom>XML et Java
    </possedeNom>
    <estEnseignant rdf:nodeID="oc"/>
    <etudiantsCours rdf:nodeID="classeIST"/>
  </Cours>
  <Cours rdf:ID="Cours2">
    <possedeNom>Ontologie conception et realisation
  </possedeNom>
  <estEnseignant rdf:nodeID="oc"/>
  <etudiantsCours rdf:nodeID="classeIST"/>
</Cours>
<Enseignant rdf:nodeID="oc">
  <possedeNom>Olivier Cure</possedeNom>
</Enseignant>
```

Solution 2 : RDF (2)

```
<rdf:Seq rdf:nodeID="classeIST">
  <rdf:li>
    <Etudiant rdf:ID="etud01">
      <possedeNom>Pierre Dupont</possedeNom>
    </Etudiant>
  </rdf:li>
  <rdf:li>

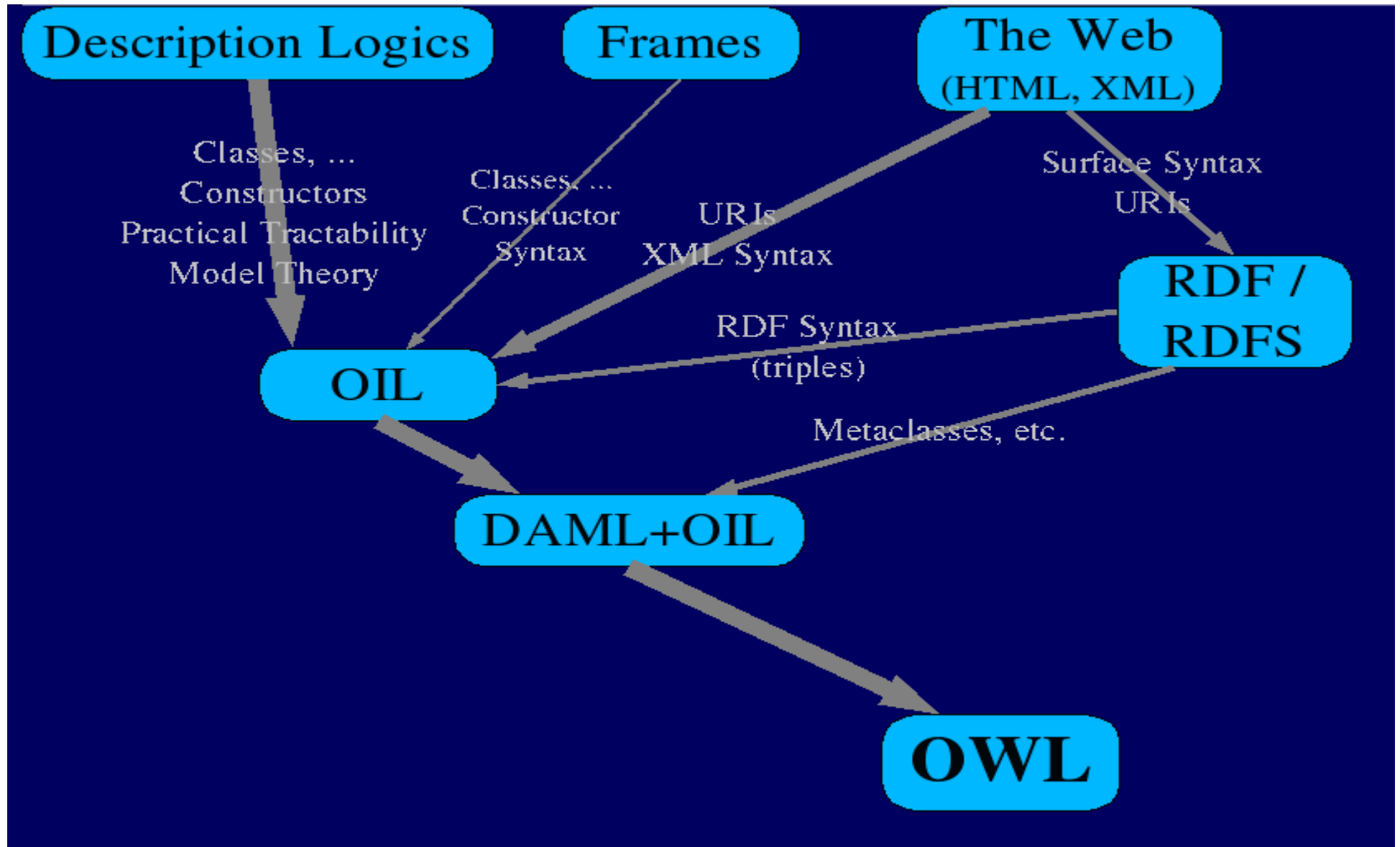
    <Etudiant rdf:ID="etud02">
      <possedeNom>Alain Durand</possedeNom>
    </Etudiant>
  </rdf:li>

  <rdf:li>
    <Etudiant rdf:ID="etud03">
      <possedeNom>Marie Martin</possedeNom>
    </Etudiant>
  </rdf:li>
</rdf:Seq>
</rdf:RDF>
```

[2.2]

Logiques de descriptions et OWL

Influences de OWL



Presentation des DL

- une famille de langages de KR
- Représentation avec une structure et une formalisation bien maîtrisée.
- Un formalisme logique pour la représentation d'information sur des concepts, des individus et leurs descriptions.
- Etude bien maîtrisée du rapport expressivité/complexité.
- Une fondation du Semantic Web avec les langages DAML+OIL, OWL.

Historique (1)

- Phase 1 (1980-1990)
 - Implémentation des systèmes comme KL-ONE, K-REP, BACK, LOOM.
 - BACK et LOOM sont relativement expressifs mais incomplets.
 - Algorithme de subsomption structurelle demande une normalisation de la description des concepts et une comparaison récursive des structures syntaxiques des descriptions.
 - Algorithme généralement efficace (polynomial) mais complet uniquement pour des DL non expressive.
 - Le système CLASSIC propose une restriction de l'expressivité pour atteindre la complétude.

Historique (2)

- Phase 2 (1990-1995)
 - Systèmes : CRACK, KRIS (expressive and complete).
 - Utilisation d'algorithmes basés sur la méthode des tableaux.
- Phase 3 (1995 – 2000)
 - DL très expressives et optimisation des systèmes (FaCT, Racer, DLP)
- Phase 4 (2000->)
 - Phase industrielle : Web Semantique, Bases de données.

Vue générale des DL

- Un langage pour exprimer des assertions factuelles, connaissance intensionnelle et des requêtes.
- Concepts – pour déclarer des entités, classes.
 - Ex : etudiant
- Rôles – pour déclarer des propriétés, des relations
 - Ex : possedeAmi
- Des constructeurs pour définir les expressions des concepts
 - Ex : $\text{Etudiant} \sqcap \exists \text{possedeAmi}.\text{Docteur}$
- Objets – instances des classes
 - Ex : pierre

Vue générale des DL (2)

- Un sous-ensemble sans les fonctions de FOL
- Avantages des DL sur FOL
 - Structuration des connaissances est plus riche => guider les inférences
 - Procédures pour les inférences plus efficaces (décidabilité).
- Que des prédicats, pas de variables explicites
- Les expressions des concepts sont utilisées pour exprimer l'information terminologique et sur les instances.
- Choix des constructeurs est la base du rapport expressivité/ complexité.

Concepts

- Des concepts définis (Defined) sont introduits en donnant toutes des conditions nécessaires et suffisantes.
- Des concepts primitifs sont introduits en donnant des conditions nécessaires.

Sémantique des constructeurs

<u>Construct</u>	<u>Syntax</u>	<u>Semantics</u>
Concept	A	$A^I \subseteq \Delta^I$
Role name	R	$R^I \subseteq \Delta^I \times \Delta^I$
conjunction	$C \sqcap D$	$C^I \cap D^I$
value restriction	$\forall R.C$	$\{x \in \Delta^I \mid \forall y.(x,y) \in R^I \Rightarrow y \in C^I\}$
existensial quantification	$\exists R$	$\{x \in \Delta^I \mid \exists y.(x,y) \in R^I\}$
top	\top	Δ^I
bottom	\perp	\emptyset
negation (C)	$\neg A \neg C$	$\Delta^I \setminus C^I$
disjunction (U)	$C \sqcup D$	$C^I \cup D^I$
existential restriction (E)	$\exists R.C$	$\{x \in \Delta^I \mid \exists y.(x,y) \in R^I \wedge y \in C^I\}$
number restrictions (N)	$(\geq n R) (\leq n R)$	$\{x \in \Delta^I \mid \#(y \mid (x,y) \in R^I) \geq n\}$
collection of individuals (O)	$\{a_1 \dots a_n\}$	$\{a_1^I \dots a_n^I\}$
Role heirarchy	$R \sqsubseteq S$	$R^I \subseteq S^I$
Inverse role	R°	$\{(y,x) \mid (x,y) \in R^I\}$
Qualified number restriction	$(\geq n R.C) (\leq n R.C)$	$\{x \in \Delta^I \mid \#(y \mid (x,y) \in R^I \wedge y \in C^I) \geq n\}$

KB, Abox et Tbox

- Une notion clé des DL : la séparation de la connaissance conceptuelle, intensionnelle ou Terminologique (Tbox) de la connaissance des instances, extensionnelle ou Assertionnelle (Abox)
 - Tbox : $\text{has_son} = \exists \text{childOf.Male}$
 - Abox : $\text{Male(John), Male(Peter), childOf(John, Peter)}$
- Raisonnement en utilisant $\text{KB} = \langle \text{Tbox}, \text{Abox} \rangle$

Tbox

- Typiquement, une Tbox contient des déclarations de la forme :
 - $\text{Woman} \equiv \text{Person} \cap \text{Female}$
 - $\text{UnderGradStudent} \subseteq \text{Student}$
 - $\text{GradStudent} \subseteq \text{Student} \cap =1 \text{ degree} \cap \forall \text{ degree.String}$
- Raisonnements disponibles :
 - Satisfaction de concepts : $\text{KB} \not\models C \equiv \perp$
 - Subsumption de concepts : $\text{KB} \models C \subseteq D$ i.e. $C \sqcap \neg D \equiv \perp$
 - Consistence : $\text{KB} \not\models$, il existe un modèle pour KB.
- Subsumption est à la base de la classification, contrôles de consistance, etc..

Abox

- Des assertions sur individuals
 - $C(a)$, $R(a,b)$
- **Instance checking** : vérifier si un individu donné est une instance d'un concept donné dans l'ensemble des modèles de KB. $KB \models C(a)$
- **Realization** : trouver pour un individu donné, le concept le plus spécifique dont il est une instance
- **Retrieval** : trouver les individus d'une KB qui sont des instances d'un concept donné.

Méthode des tableaux

- Algorithme des tableaux pour tester la satisfiabilité.
 - Construction d'un modèle
 - Le modèle est représenté sous la forme d'un arbre T où :
 - Les noeuds de T correspondent à des individus.
 - Les noeuds sont labellisés avec les sous-concepts de C
 - Les arcs sont labellisés avec les relations sur C

Example

- Définir la terminology suivant les caractéristiques suivantes :
 - A man is a human
 - A woman is a human
 - No man is a Woman, and vice-versa.
 - A team is defined as the set with at least 2 members which are all humans.
 - A small team is defined as a team with at most 5 members.
 - A modern team is defined as a team with at most 4 members and with at least 1 leader, which is a member, and all leaders are women.

Formalization in DL

$Human \sqsubseteq Top$

$Man \equiv Human$

$Woman \equiv Human \sqcap \neg Man$

$Team \equiv \forall member. Human \sqcap \leq 2 member$

$SmallTeam \equiv Team \sqcap \geq 5 member$

$leader \sqsubseteq member$

$ModernTeam \equiv Team \sqcap \leq 1 leader \sqcap \forall leader. Woman$

Examples

$BusDriver \equiv Person \sqcap \exists drives.Bus$

$Driver \equiv Person \sqcap \exists drives.Vehicule$

$Bus \sqsubseteq Vehicule$

- A BusDriver is a subclass of Driver.

$CatOwner \equiv Person \sqcap \exists hasPet.Cat$

$hasPet \sqsubseteq likes$

$Catliker \equiv Person \sqcap \exists likes.Cat$

- A CatOwner is a subclass of CatLiker.

Examples (2)

$Driver \equiv Person \sqcap \exists drives.Vehicule$

$Driver \sqsubseteq Adult$

$GrownUp \equiv Adult \sqcap Person$

- A Driver is a GrownUp

$OldLady \equiv Person \sqcap Elderly \sqcap Female$

$OldLady \sqsubseteq \forall hasPet.Cat \sqcap \exists hasPet.Animal$

$CatOwner \equiv Person \sqcap \exists hasPet.Cat$

- OldLady is a CatOwner

OWL

- Une recommandation du W3C pour représenter des ontologies pour le Web Sémantique.
- Influences : RDF, DL et les frames
- OWL est un langage pour structurer l'information dans des ontologies : définir les concepts d'un domaine et les relations liant ces concepts.
- OWL permet également de définir les instances.

OWL

- OWL possède 3 sous-langages : OWL Lite, OWL DL and OWL Full
- OWL propose plusieurs syntaxes : RDF, OWL/RDF, abstract syntax.
- Termes utilisés avec OWL :
 - Concepts = classes
 - Rôles = propriétés

OWL comme une DL

OWL Abstract Syntax	DL Syntax
intersectionOf	$C_1 \sqcap \dots \sqcap C_n$
unionOf	$C_1 \sqcup \dots \sqcup C_n$
complementOf	$\neg C$
oneOf	$\{a_1\} \sqcup \dots \sqcup \{a_n\}$
allValuesFrom	$\forall P.C$
someValuesFrom	$\exists P.C$
maxCardinality	$\leq n P$
subClassOf	$C_1 \sqsubseteq C_2$
equivalentClass	$C_1 \equiv C_2$
disjointWith	$C_1 \sqsubseteq \neg C_2$
sameIndividualAs	$\{a_1\} \equiv \{a_2\}$

Syntaxe

- OWL/RDF syntax

`<owl:Class rdf:ID="Cat">`

`<rdfs:subClassOf rdf:resource="#Animal"/>`

`</owl:Class>`

- Abstract syntax

`Class (a:Cat partial a:Animal)`

OWL Lite

- OWL Lite ressemble aux Frames : supporte la classification hiérarchique des classes et propriétés, des contraintes simples
- OWL Lite possède moins de constructeurs que OWL DL
 - Descriptions dans des descriptions
 - Unions, OneOf, etc..
- OWL Lite correspond à DL SHIN(Dn)
 - Qui est décidable
 - Avec des implémentations efficaces.

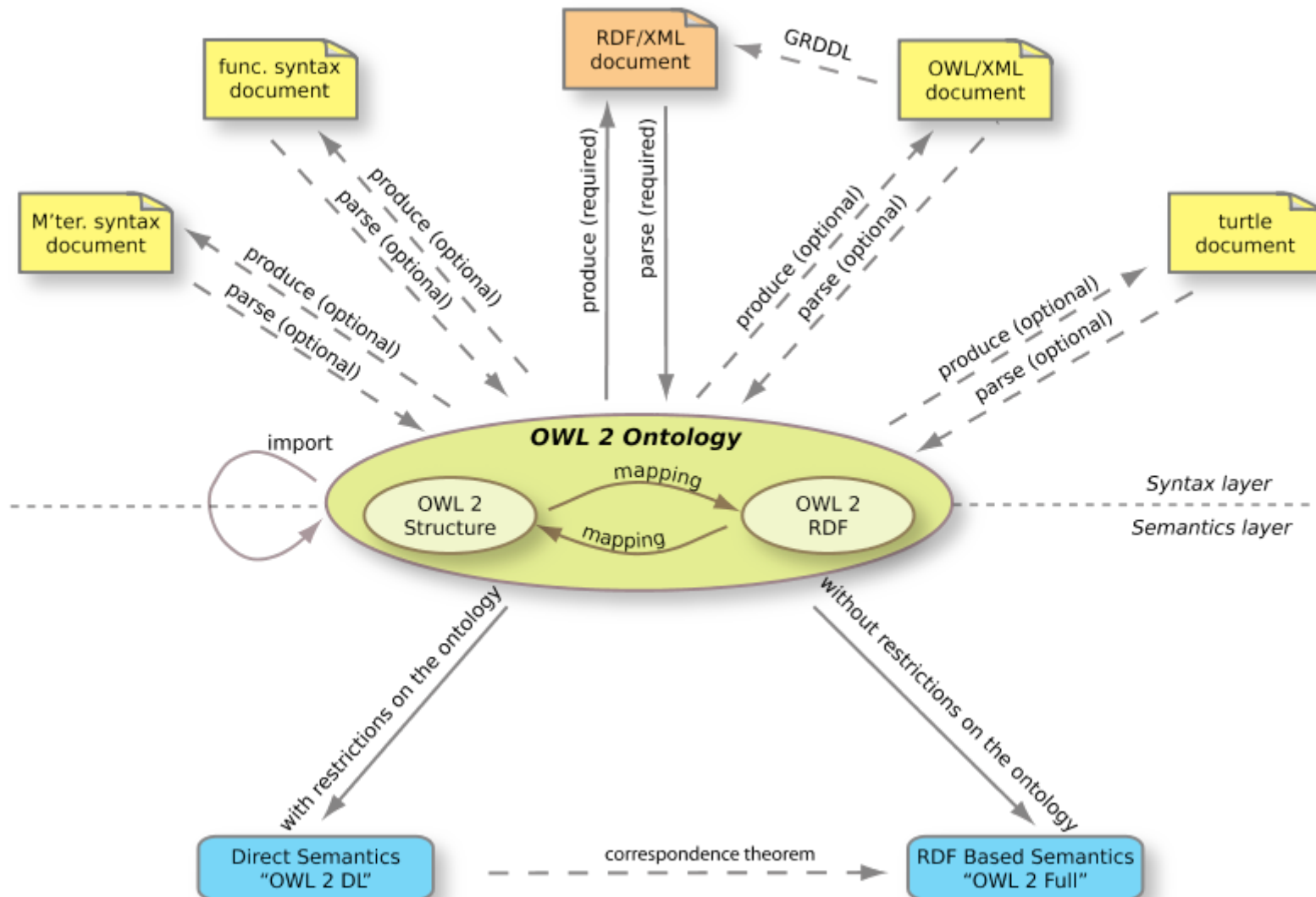
OWL DL

- Expressivité riche en gardant la complétude computationnelle et la décidabilité.
- Ne permet pas toute la liberté syntaxique de RDF :
 - Ex : une classe ne peut être une propriété ou une instance, une propriété ne peut être une classe ou une instance.
- Correspondence avec la DL SHOIN(D)

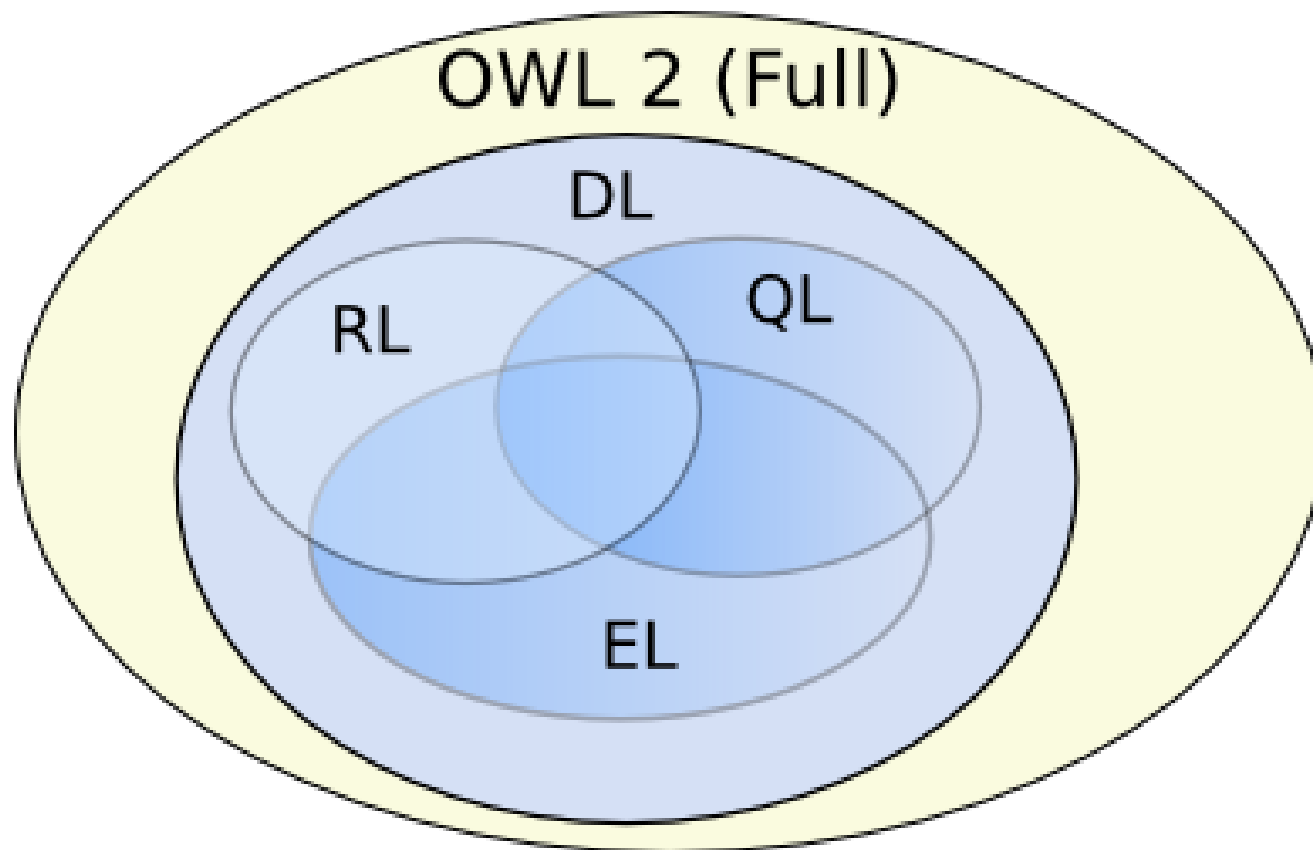
OWL Full

- Très expressif (méta-classes, classes comme valeur d'object property).
- Profite de toute la liberté syntaxique de RDF
 - Une classe peut être traitée comme un ensemble d'individus et comme une classe.
- Pas d'algo efficace pour le raisonnement.

OWL 2



OWL 2 profiles



[3]

Les outils

[3.1]

Editeur : Protégé

Autres solutions: OilEd, KAON, SWOOP, OntoEdit, Ontolingua, OntoSaurus, WebODE, WebOnto

Protégé

- Un éditeur d'ontologies et de bases de connaissances
- Open-source
- Disponible à l'URL :
<http://www-protege.stanford.edu/>
- Développé à l'Université de Stanford en Java
- Supporte de nombreux plug-ins.

Historique

- Début des années 90 : Protégé II
 - Environnement pour l'ingénierie des connaissances permettant de définir des modèles et de générer des GUI. Version pour NeXTSTEP.
- Milieu des 90s : ProtégéWin
 - Version pour Windows
- Fin des 90s : Protégé2000
 - Version Java, Open-source
 - Développement de plug-ins
- 2003 : Protégé (à partir de la version 2.0)
 - Support de OWL.

Plateforme pour plug-ins

- Communauté très active. On trouve des plug-ins pour :
 - La visualisation
 - L'inférence
 - Import et export de formats (XML, RDF, DAML+OIL, OWL, CLIPS, Topic Maps).
 - Conception d'IHM
 - etc..

Démonstration

- Créer des classes, des propriétés, des instances
- Lancer la classification avec Pellet.
- Ecrire et exécuter des requêtes en SPARQL.
- Visualisation de graphes

[3.2]

Framework Java : Jena

JENA : Présentation

- Plateforme Java pour le Web sémantique
- Open-source
- Développé par un laboratoire de Hewlett-Packard

JENA : URL

- La page d'accueil de JENA :
 - <http://jena.sourceforge.net/downloads.html>
- Un groupe de discussions :
 - <http://groups.yahoo.com/group/jena-dev/>
- La javaDoc
 - <http://jena.sourceforge.net/javadoc/index.html>

JENA : éléments

- Un analyseur RDF : ARP – Another RDF Parser
- un langage de requêtes pour RDF : RDQL maintenant SPARQL
- Persistence des données, en particulier avec les SGBD (Oracle, MySQL, PostgreSQL)
- Support de RDF, RDFS, OWL
- Inférence

Modèle

- L'architecture de JENA est centrée sur la notion de modèle (*model*) : l'ensemble des déclarations qui composent un document, graphe ou instantiation d'un vocabulaire.
- A partir de JENA2.0, on doit créer un modèle en s'aidant de ModelFactory.

ModelFactory : code

```
import java.util.Iterator;
import com.hp.hpl.jena.rdf.model.*;
public class EtudRDF {
    private String etudNS = "file:/home/olive/_mesCours/
kr/etudIns2.rdf";
    public static void main(String[] args) {
        EtudRDF etudRDF= new EtudRDF();

        etudRDF.load();
    }
    public void load() {
        Model model = ModelFactory.createDefaultModel();

        model.read(etudNS);

        model.write(System.out);
    }
}
```

Création / Enrichissement d'un modèle (1)

```
model = ModelFactory.createDefaultModel();
jean = model.createResource(familleURI+"jean");
marie = model.createResource(familleURI+"marie");
dominique = model.createResource(familleURI+"dominique");
alexandra = model.createResource(familleURI+"alexandra");
baptiste = model.createResource(familleURI+"baptiste");
pierre = model.createResource(familleURI+"pierre");

enfantDe = model.createProperty(relationshipURI, "childOf");
parentDe = model.createProperty(relationshipURI, "parentOf");
epouseDe = model.createProperty(relationshipURI, "spouseOf");

dominique.addProperty(parentDe, baptiste);
dominique.addProperty(parentDe, pierre);
alexandra.addProperty(parentDe, baptiste);
alexandra.addProperty(parentDe, pierre);
alexandra.addProperty(epouseDe, dominique);
```

Création / Enrichissement d'un modèle (2)

```
Statement statement = model.createStatement(dominique,
    enfantDe, jean);
model.add(statement);
statement = model.createStatement(dominique, enfantDe,
    marie);
model.add(statement);
```

```
Property prop =
    model.createProperty(relationshipURI, "knowsOf");
//Création d'un noeud vide
Resource blank = model.createResource( )
    .addProperty(prop, "personne1")
    .addProperty(prop, "personne2")
    .addProperty(prop, model.createLiteral("personne3", "fr"));

// Affichage du document RDF au format RDF/XML (par défaut)
model.write(new PrintWriter(System.out));
```

Accès aux données

- On peut accéder aux données :
 - à l'aide de programmes en s'appuyant sur les méthodes et classes de l'API
 - À l'aide de langage de requêtes SPARQL

Via les données : code

```
// Ensemble des déclarations  
StmtIterator iter = model.listStatements();  
while (iter.hasNext()) {  
    Statement stmt = (Statement) iter.next();  
    System.out.println(stmt.getSubject()+"-  
    (" +stmt.getPredicate()+" ) -  
    >" +stmt.getObject().toString());  
}
```

Via les données : code (2)

```
// Connaitre les parents
ResIterator parents =
    model.listSubjectsWithProperty(parentDe);
while (parents.hasNext()) {
    personne = parents.nextResource();
    System.out.println(personne.getLocalName() + " -> URI
        =" + personne.getURI());
}
// Encore des parents avec enfantDe
NodeIterator parentsSuite =
    model.listObjectsOfProperty(enfantDe);
while (parentsSuite.hasNext()) {
    Resource person = (Resource) parentsSuite.nextNode();
    System.out.println( person.getLocalName() + " -> URI
        =" + person.getURI());
}
```


Via les données : code (3)

```
System.out.println("Les ens :");
NodeIterator itNode =
    model.listObjectsOfProperty(estEns);
while (itNode.hasNext()) {
    RDFNode node = (RDFNode) itNode.next();
    System.out.println("Enseignant
        =" + node.toString());
    // on veut le nom
    Property nom =
        model.createProperty("http://www.univ-
            mlv.fr/~ocure/etudSchema.rdf#possedeNom");
    NodeIterator itNom =
        model.listObjectsOfProperty((Resource) node, nom);
    while (itNom.hasNext()) {
        System.out.println("Nom = " + itNom.next());
    }
}
```

[3.3]

Raisonneur : Pellet

Autres solutions: RacerPro, FACT, KAON, Hermit,
Cerebra

Pellet

- Un raisonneur pour OWL2 DL
- Basé sur la méthode des tableaux avec de nombreuses optimisations et fonctionnalités innovantes:
 - Raisonnement sur les types de données
 - Analyse et réparation d'ontologies (OWL DL et Full)
 - Debugging d'ontologies
 - Raisonnement incrémental
 - Support SWRL (règles)

Utilisation

- Programmation en Java au travers des API:
 - Interconnection avec les API Jena, OWLAPI
- Ligne de commande
- Port http (démonstration)

[3.4]

Requêtes : SPARQL

Autres solutions: RDQL, SQUISH, SeRQL, ..

Introduction

- Langage de requêtes pour données RDF
- Principe: graph pattern matching
- Principales clauses:
 - SELECT, CONSTRUCT, DESCRIBE, ASK
- Avec filtre pour restreindre les valeurs.
- Options:
 - ORDER BY, LIMIT/OFFSET, DISTINCT, REDUCED

Motifs (patterns)

- Variables are prefixed with a '?'
- Patterns use triple forms
- Example:
SELECT ?s ?p ?o
WHERE {?s ?p ?o}
- Motif comme conjunction of triplets:
{?x rdf:type ex:Person.
?x ex:nom ?name}

Example

```
@prefix person: <http://example/person/> .  
@prefix foaf: <http://xmlns.com/foaf/0.1/> .  
person:A foaf:name "Alice" .  
person:A foaf:mbox <mailto:alice@example.net> .  
person:B foaf:name "Bob" .
```

```
PREFIX person: <http://example/person/>  
PREFIX foaf: <http://xmlns.com/foaf/0.1/>  
SELECT ?name  
WHERE { ?x foaf:name ?name }
```

name

"Bob"

"Alice"

Example (2)

```
@prefix person: <http://example/person/> .  
@prefix foaf: <http://xmlns.com/foaf/0.1/> .  
person:A foaf:name "Alice" .  
person:A foaf:mbox <mailto:alice@example.net> .  
person:B foaf:name "Bob" .
```

```
PREFIX person: <http://example/person/>  
PREFIX foaf: <http://xmlns.com/foaf/0.1/>  
SELECT ?name  
WHERE { ?person foaf:mbox  
        <mailto:alice@example.net> .  
        ?person foaf:name ?name . }
```

name

"Alice"

Example (3a)

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .  
@prefix stock: <http://example.org/stock#> .  
@prefix inv: <http://example.org/inventory#> .  
stock:book1 dc:title "SPARQL Query Language Tutorial" .  
stock:book1 inv:price 10 .  
stock:book1 inv:quantity 3 .  
stock:book2 dc:title "SPARQL Query Language (2nd ed)" .  
stock:book2 inv:price 20 ; inv:quantity 5 .  
stock:book3 dc:title "Moving from SQL to SPARQL" .  
stock:book3 inv:price 5 ; inv:quantity 0 .  
stock:book4 dc:title "Applying XQuery" .  
stock:book4 inv:price 20 ; inv:quantity 8 .
```

Example (3b)

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX stock: <http://example.org/stock#>
PREFIX inv: <http://example.org/inventory#>
SELECT ?book ?title
WHERE {
  ?book dc:title ?title . ?book inv:price ?price . FILTER ( ?price < 15 )
  ?book inv:quantity ?num . FILTER ( ?num > 0 ) }
```

book	title
------	-------

stock:book1	"SPARQL Query Language Tutorial"

Example (4)

```
@prefix person: <http://example/person/> .
```

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
```

```
person :a foaf:name "Alice" .
```

```
person :a foaf:nick "A-online" .
```

```
person:b foaf:name "Bob" .
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
```

```
SELECT ?name ?nick
```

```
{ ?x foaf:name ?name .
```

```
  OPTIONAL { ?x foaf:nick ?nick } }
```

```
name | nick
```

```
-----  
"Alice" | "A-online"
```

```
"Bob" |
```

[3.5]


Triple store : Sesame

Autres solutions: Oracle, Virtuoso

Sesame

- Un framework pour le stockage, l'exécution de requêtes et inférence sur des données RDF et RDFS (SPARQL, SeRQL)
- Une librairie Java pour manipuler RDF
- Un server de base de données pour accéder à des répertoires de données RDF

Sesame

 **Workbench**

OpenRDF

Sesame server
Repository
Modify
[Query](#)
Explore
Extract
System

Current Selections
Sesame server: <http://132.180.195.113/openrdf-sesame/> [\[change\]](#)
Repository: DaltOn (DaltOn) [\[change\]](#)

SELECT Query CONSTRUCT Query Boolean (ASK) Query

Query

SELECT Query
Query Language: SPARQL ▼

Query:

```
select ?x ?y WHERE
{ ?x <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.univ-mlv.fr/~ocure/ontologies/refMed.owl#Drug> }
```

☐ Include inferred statements

Submit

Copyright [Aduna](#) © 1997-2009

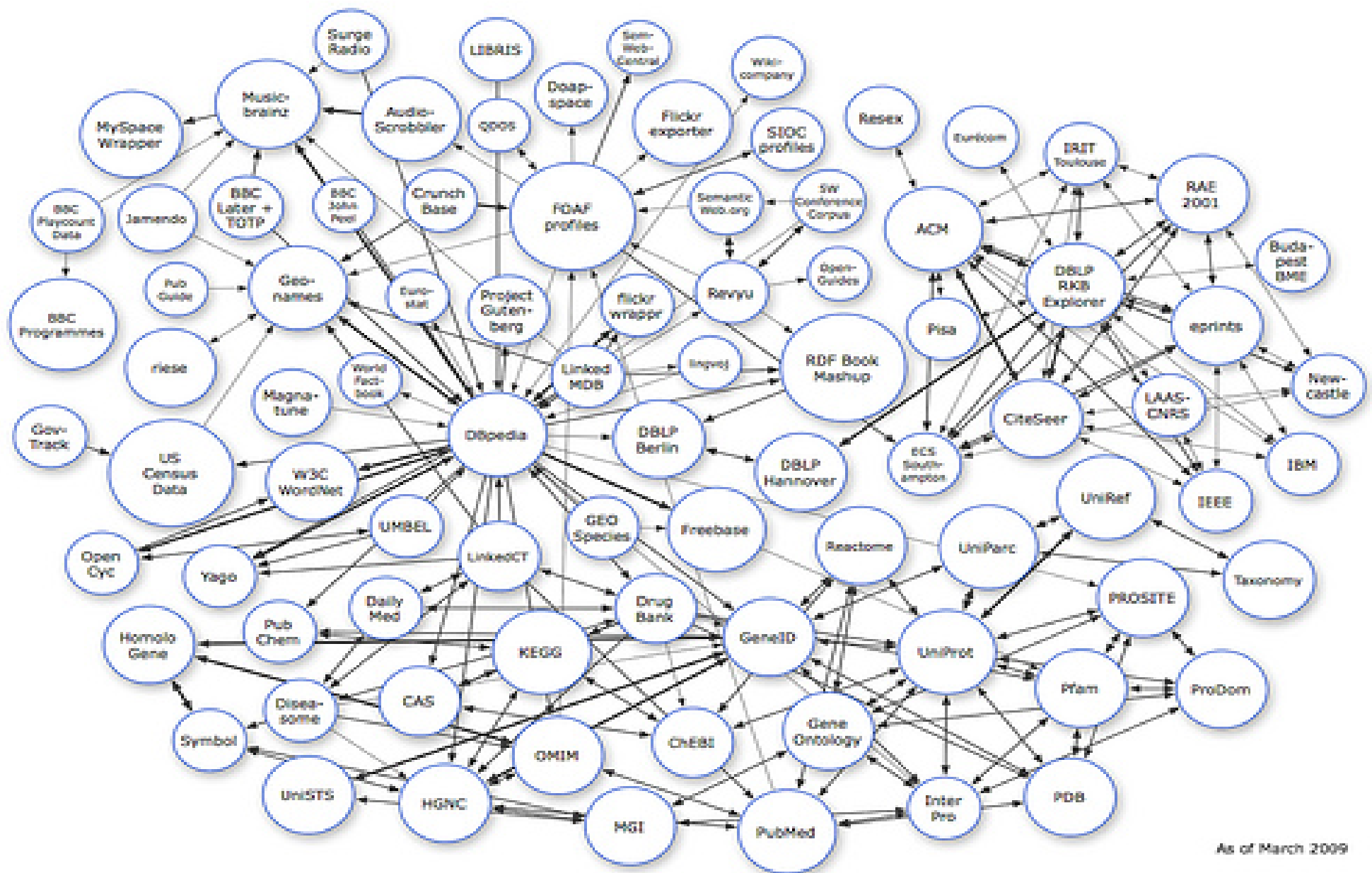
Conclusion

- Déjà beaucoup d'outils (open source)
- Vitrine: Réseaux sociaux (FOAF), OpenLinking data.
- Yahoo! Intègre le RDF.
- Montée en charge au niveau des données du Web?

Chantier

- OWL2
 - Relations avec les bases de données
 - Interaction avec un système de règles
- Services Web sémantiques
- Semantic Web et Peer to Peer

Merci



As of March 2009